Numerical Heat Transfer, Part B, 41: 211–237, 2002

1040-7790/02 $12.00 + .00

CURRENT VIEWS ON GRID GENERATION: SUMMARIES OF A PANEL DISCUSSION

Rod W. Douglass
Computational Science Methods Group, Los Alamos National Laboratory, Los Alamos, New Mexico, USA

Graham F. Carey
Department of Aerospace Engineering and Engineering Mechanics, The University of Texas at Austin, Austin, Texas, USA

David R. White
Parallel Computing Sciences Department, Sandia National Laboratories, Albuquerque, New Mexico, USA

Glen A. Hansen
Computational Science Methods Group, Los Alamos National Laboratory, Los Alamos, New Mexico, USA

Yannis Kallinderis
Department of Aerospace Engineering and Engineering Mechanics, The University of Texas at Austin, Austin, Texas, USA

Nigel P. Weatherill
Faculty of Engineering, University of Wales Swansea, Swansea, United Kingdom

This article summarizes presentations made by the panelists forming the Panel Session on Grid Generation held at the Second International Symposium on Advances in Computational Heat Transfer (CHT'01) in May 2001 at Palm Cove, Queensland, Australia. First a brief summary of the grid generation process is presented by R. Douglass. G. Carey presents an assessment of grid generation and associated issues, trends, and techniques. D. White provides an overview of hexahedreal meshing, followed by a discussion of adaptive mesh refinement methods by G. Hansen. Y. Kallideris gives an update on hybrid grid methods for Navier–Stokes problems, and N. Weatherill concludes with a look forward in a discussion of large-scale simulations on unstructured grids.

Received 23 July 2001; accepted 5 October 2001.

The research presented in Section 2 was performed under the auspices of the U.S. Department of Energy under contract W-7405-ENG-36. The research presented in Section 3 has been supported in part by the U.S. Department of Energy's Accelerated Strategic Computing Initiative grant B347883. The research presented in Section 5 was performed under the auspices of the U.S. Department of Energy under contract W-7405-ENG-36. The authors of Section 7 wish to thank the European Union for funding the JULIUS project (Esprit 25050), under which some of this work was undertaken. B. Larwood would also like to thank EPSRC for financial support. The aerospace-duct test case was kindly provided by BAE Systems, Sowerby Research Centre.

Address correspondence to R. W. Douglass, Computational Science Methods Group, MS F645, Los Alamos National Laboratory, Los Alamos, NM 87545, USA. E-mail: rwd@lanl.gov

NOMENCLATURE

ξ, η	coordinates in logical smoothing space	$\mathbf{x}$	nodal coordinate locations in 2-space (x, y)
g_{ij}	surface metric tensor, $i, j = 1, 2$		

1. INTRODUCTION

As computational heat transfer matures, the problems being solved are increasingly complex. Complexity of the simulation may, for example, revolve around geometric details in three dimensions, the physical properties of the materials to be simulated, or the physical phenomena of the simulation such as reacting flows, thermal radiation, turbulence, or combinations thereof. From a grid-generation perspective, these complexities lead to increasing difficulties in generating computational grids based on a faithful adherence to physical geometric material boundaries which may have been obtained directly from computer-aided design (CAD) drawings, in resolving three-dimensional engineering details such as bolts or screws or piping discontinuities, or in using grid quality measures and methods that dynamically preserve some appropriate form of global or local numerical accuracy. One easily is led to the conclusion that grid generation for such problems is now much more than simply N do-loops for N-dimensioned-space problems.

The Second International Symposium on Advances in Computational Heat Transfer (CHT'01) was viewed as an ideal forum for grid-generation issues and, consequently, the organizers agreed to offer a panel discussion highlighting the importance of grid generation to simulation success and to provide an opportunity for dialog on the theme of grid generation. This article is a summary of the views presented by the panelists at that session.

2. THE GRID-GENERATION PROCESS: CONTRIBUTED BY R. DOUGLASS

2.1. Introduction

When a (partial) differential equation set is to be solved in the context of a set of given physical properties within a given geometric domain, either of which make it intractable to solve exactly, approximate methods of solution are used. The approximate, discrete methods used in computational heat transfer are usually either the structured finite-difference methods or discrete weighted-residual methods such as finite-volume or finite-element methods. These methods have been heavily researched over the past 30 or more years and may be viewed as being at a fair level of maturity. Approximate methods like these require a decomposition of space into discrete, contiguous volumes defining a computational grid for the problem at hand. The process used to perform this discretization, called grid generation, is not at the same level of maturity as the approximate solution methods, especially for three-dimensional problems.

2.2. The Process

From a grid-generation perspective, grid generation begins with a faithful representation of the geometry of the physical problem being modeled. Geometric information may be rather straightforwardly implemented (e.g., modeling flow in a uniform-diameter straight pipe), or it may be so complex that it comes as a multipart object from a CAD system (e.g., flow and heat/mass transfer in the cylinder head region of an internal combustion engine). There are typically three ways used to represent geometry [1]: constructive solid geometry (CSG), boundary representation (b-reps), and domain decomposition representation (dd-reps). CSG defines a domain using a formula of set theory operations over a collection of primitive geometric shapes (spheres, boxes, cones, etc.). B-reps define the geometry in terms of piecewise low-degree polygons fitted to its boundaries, such as a nonuniform rational B-spline (NURBS) representation as from a CAD or CSG model. Finally, dd-reps describe the geometry of the boundaries by a discretization into nonoverlapping polyhedrea, again from CAD or CSG models. Any flaws in the geometry must be repaired prior to grid generation, a process termed geometric healing. Healing is often required for geometry obtained from a CAD model.

The geometric difficulty in terms of grid generation lies in the inherent discrete (piecewise) approximation of the actual domain geometric boundaries and their included material regions, due to the decomposition of the physical domain into linear or low-order polynomial volumes such as triangles, squares, tetrahedrons, hexagons, etc. This is called discretization. The faces of the discrete volumes, often called cells, elements, or zones, are not in general boundary-fitted. That is, they would ideally be locally conforming to the exact geometry or boundary, thereby being boundary-fitted faces and, thus, boundary-fitted cells. The result of such a discretization would be a faithful representation of the actual physical domain. To the extent that the element faces are not boundary-fitted, the errors in the discrete approximation must also include a measure of the geometry (or actually physical material) *lost or gained* in the boundary approximation.

There are additional concerns in the discretization process other than geometric. The type of discretization used, that is, the sort of element shapes and sizes used, depends on the material properties within and across material boundaries, whether the physical processes being simulated have preferred directions, whether the problem must be decomposed for parallel processing, the desired local and global accuracy measures of the approximate method, and so forth. Of course, at the time of the initial discretization, the grid may very likely have no idea as to the future presence of a shock, for example, or in which direction it is propagating.

The third part of the grid-generation process involves postgeneration modifications to the grid to improve its *quality*. Quality measures may include combinations of geometric ideas such as orthogonality of the grid, the uniformity of volumes in the grid, and the relative amount of grid skewness or nongeometric ideas such as error measures associated with the discretization of the dependent variables in the problem. Quality modifications to the grid may be a one-time event done prior to submitting the grid to the equation solver, or may be a dynamic process either done explicitly for each step in the solver (iteration or time step) or implicitly linked to the physics variables to be solved for, all as one large problem—the grid along

with the physics. Modifications may also require local refinement or derefinement of the grid in order to meet the grid quality measures selected for the problem at that current time step.

Each of these three pieces of the grid-generation process—geometry, discretization, and grid modification—is a challenging area of research. Some advances within each have been made, but the challenge remains to present users with a unified grid-generation tool set incorporating even some of the ideas listed above.

2.3. Basic Concepts

Categories of grids. Grids are either structured or unstructured. A structured grid is characterized by all grid elements being similar in shape (e.g., rectangles or hexahedra), as in a Cartesian grid (an i, j, k grid), the grid is easy to generate and has a simple data structure, by grid quality deterioration as domain complexity increases, and by strong performance for problems with smooth solutions. An unstructured grid, however, has varying element topology and size (e.g., mixtures of hexahedral and tetrahedral elements), it is more difficult to generate these grids, and they require a more complicated data structure, typically with each element being composed of a list of faces that themselves are composed of lists of nodes. The grid may then be connected through, for example, a graph of all neighboring nodes for each node. As domain complexity increases, the grid quality tends to remain high, and they perform well on problems with nonsmooth solutions such as shocks or combustion fronts.

Basic grid-generation methods. There are several algorithms or methods developed for generating grids [1], some of which are well researched and others of which are still under development. Delaunay triangulation is discussed in Section 3, while quadtree and octree refinement are discussed in Sections 5 and 6. Sphere packing, advancing-front, and methods still under development, namely, medial surface/potential methods, whisker weaving, h-morph, hexahedral-tetrahedral plastering, etc., are outlined in Section 4.

Grid quality concerns. Once a grid is established, it may be necessary to modify it due to locally or globally poor quality. Grid quality depends, of course, on the measure used to quantify quality. Such a measure might include some or all of: degree of capture of the relevant geometric details of the physical domain as well as the length scales of the physics being simulated, the level of grid conformality, smoothness, alignment with physical phenomena, aspect ratio, and orthogonality. These ideas have been used recently to optimize three-dimensional grids within a given grid topology (cf. Section 5).

Of course, there remain many open questions as to optimization of grid quality, one of which is the mathematical relation between truncation error for the discrete formulation of the governing equations and the metric elements listed just above. Another is the advantage of directly coupling the grid quality optimization process with the physics solution, thereby solving for grid point locations at the same time as the physics using an appropriate metric to couple the grid and the physics. What, then, are the best metrics? What topological operations beyond refinement will be needed to produce the optimum simulation? If grid point movement is explicitly uncoupled from the physics solve, then how should the underlying physics be

remapped onto the new grid in order to conserve mass, momentum, energy, etc.? For parallel processing, how is dynamic grid refinement done?

3. GRID GENERATION, GRID MANIPULATION, RELIABILITY, AND ACCURACY: TECHNIQUES, TRENDS, AND OPEN ISSUES: CONTRIBUTED BY G. F. CAREY

3.1. Introduction

The rapid development of microelectronics during the past three decades, and continuing hardware advances, have dramatically expanded our computational capabilities. John von Neumann remarked in the middle of the last century, in his report on the ENIAC and simulation work related to the Manhattan Project, that computer technology would impact our entire approach to engineering and scientific analysis. He mentioned specifically such areas as fluid dynamics and electromagnetics but pointed out that the influence would be pervasive. The growth in computer technology has exceeded our wildest expectations, and we are now in a position to address complex nonlinear applications in engineering and science in an unprecedented manner. Moreover, the relatively recent advances in parallel PC cluster technology imply that very significant computational power will be widely available and inexpensive.

From the standpoint of engineering analysis and design in areas such as coupled fluid flow and transport, this implies that complex engineering systems can now be analyzed and redesigned using optimization strategies or controlled by optimal control approaches. It also implies that many problems involving nonlinear, coupled multiphysics and multiscale behavior are now within reach of computer simulation for the first time. What, then, remain as major impediments to progress? Perhaps the key obstacles are not in the analysis methodology and computational kernels, but in our ability to generate and adapt the underlying unstructured grids in a way that will yield reliable accurate simulations. The nature of the problem we encounter here is best illustrated by noting that a real engineering application to heat transfer, fluid flow, and stress analysis in a system involving hundreds of components may take several man-weeks to grid using the software tools available today. Indeed, even for quite simple domains, present state-of-the-art grid generators may fail to complete a valid grid or may complete a grid containing cells that are ill-shaped or otherwise unacceptable.

The goal here is to summarize briefly some of the key ideas associated with unstructured grid generation and grid adaption, describe current progress in these areas, and state some open issues or problems that warrant attention. We begin with some comments on automated unstructured grid generation, and cell quality, discuss related issues for grid smoothing and adaptation, include remarks on the role of error indicators in ascertaining computational reliability of a grid, and conclude with some additional observations arising out of the load-balancing needs for parallel computing.

3.2. Unstructured Grid Generation

Techniques of general applicability for automated grid generation fall mainly into two categories: (1) Delaunay schemes and (2) advancing-front algorithms. In the

Delaunay approach the basic idea is: given a set of points in the plane, a triangulation of the convex hull of these points can be ''improved'' by edge ''swaps'' to achieve a triangulation in which the minimum angle has been maximized. The idea is easily illustrated by considering an adjacent pair of triangles defining a convex quadrilateral—then one exchanges the shared edge (a diagonal of the quadrilateral) for the other diagonal if this yields a new pair of triangles whose minimum angle is greater than the minimum angle of the previous triangulation. This test can be conveniently implemented on a computer by testing whether the remaining vertex of the quadrilateral is inside the circumcircle defined by the other three vertices of a constituent triangle. The approach will efficiently yield 2-D triangulations that are optimal in the sense stated. The triangulation is unique within trivial edge swaps.

Some key issues in triangulation. The max–min angle property and circumcircle test equivalence does not hold in three dimensions, and face swaps in three dimensions may not reproduce the same number of tetrahedra. The max–min angle criterion is appealing in an esthetic sense, but in computations where boundary or interior layers are present, ''anisotropic'' grids that contain slender elements with small angles in these layer regions will be desirable. Likewise, in adaptive mesh refinement (AMR) or mesh redistribution, directional refinement is again desirable for these types of applications. In the spirit of the second point just made, interpolation estimates from approximation theory and due consideration of the condition number of the algebraic systems that are generated both imply that the maximum obtuse angles rather than the minimum acute angles are a more critical concern.

There are a number of software systems available for advancing-front grid generation (CUBIT Mesh Generation Toolkit, Web site: http://endo.sandia.gov/SEACAS/CUBIT/Cubit.html). These algorithms are also sometimes used to generate an initial tessellation prior to applying a Delaunay procedure or a grid-smoothing strategy. However, the strong interest expressed by the engineering community for quality meshes comprised only of hexahedra places a difficult constraint on the meshing problem in three dimensions. The essential idea here is to ''plaster'' hexahedra with quadrilateral faces on the surfaces and interfaces layer by layer so that the domain is progressively filled, but several difficulties may arise, especially for complex geometry.

Some key issues in advancing-front methods. The advancing fronts from multiple surfaces intersect and the algorithm has to be able to contend with the unusual topologies arising from these intersecting surfaces. Present approaches cannot guarantee completion of an all-hex grid in this manner. One can generate all-hex meshes from a tetrahedral grid by subdividing all the tetrahedra to hexahedra in a consistent manner, but the resulting hex cells are not well shaped and not amenable to smoothing. Hybrid grids containing hexahedral and tetrahedral elements with transition pyramids appear a viable solution, but the engineering analyst must produce stable elements (nonlocking, no hour-glassing, no spurious modes in the application class) for the base elements and the transition elements. The geometry dictates the mesh gradation in the basic advancing-front schemes, and this may not be the grading desired by the analyst for the problem class. The problem of directional mesh grading remains.

3.3. Adaptive Refinement/Coarsening

One of the great promises in the meshing area is that of automating not only an initial grid generation but also subsequent optimization and control of the grid to tailor it to the problem being solved [2]. These ideas of adaptive control of grids are discussed in [3] and will not be elaborated upon here. Suffice it to say that much of the burden of grid-generation completion can be shouldered by means of a good adaptive refinement algorithm. Since refinement and derefinement are also logical strategies that should sensibly be part of the engineering analysis algorithm, this is the ideal approach. It also provides a convenient data structure for "mining" simulation results and for fast remote visualization. Why, then, are adaptive mesh refinement (AMR) approaches not more widely embraced and adopted as an integral part of grid-generation software and analysis software? There are several reasons. Most notably, first of all, AMR usually presumes the existence of a zero-level grid, so the main problem of grid generation may not be circumvented by AMR (although we can build octree "coverings" to include generation step); second, there is the considerable complexity of AMR data structure and programming; third, most "legacy" analysis codes are not equipped to take advantage of AMR, so the added complexity and data structures would not be exploited. Nevertheless, it is clear that AMR should be embedded in the grid-generation process and intimately tied to the analysis step in future-generation complex analysis software for complex applications. This will probably occur in industry when market share is being taken by new codes that offer these capabilities. In the national research sector it will occur when these capabilities permit analysis of problems that cannot be addressed by existing methods in a timely and economic way. This will come.

Some key issues. What AMR approaches and data structures are needed in conjunction with the grid-generation problem and the range of analysis problems? Derefinement involving reconstitution of subdivided cells is not as straightforward as refinement when one considers error control, the treatment of moving boundaries, and several other "details." Coarsening below the zero-level base grid raises an entire slew of additional problems. There are several open issues related to, for instance, the treatment of geometry representation during AMR, a-posteriori error estimates, and error indicators.

3.4. Mesh Quality and Reliability

The goal in unstructured grid generation is to provide a quality grid capable of representing the geometry and of providing the basis for a reliable accurate simulation. In general, neither of the above is ensured. This is largely because the industry does not insist on implementing appropriate tests to ensure that these goals are met. For example, in the past it has been rare indeed for industrial grid generators to include algorithms that have some form of quality assessment—usually we are relieved to get a reasonable grid that does not have any visually obvious problems such as unintended "holes" or cell overlap. If obvious faults arise during the analysis step, then the cause is frequently identified as "an incorrect or inadequate grid." In fact, the cell quality may even be excellent but the mesh resolution inadequate to resolve layers or may lead to "false solutions." Smoothing to improve mesh quality is also important [4].

Some key issues. Cell shape quality needs to be quantified, but the various metrics currently in use are duplicative, insufficient, and not yet adequately analyzed [5]. The effect of cell imperfection on the accuracy of a simulation is not yet resolved. Directional grading refinement for more efficient simulation of a specific problem violates the esthetic principles currently accepted as underlying a "good grid." A good grid for one problem will be a poor grid for another problem in the same applications class. This consolidates the argument for AMR. The ideal grid from an error analysis standpoint may be unsuitable, because of system conditioning, for computations on a computer of a given finite precision. Libraries of error indicators that diagnose possible cell quality problems and also provide indicators for simulation solution quality are needed. A concerted effort in this direction would appear to have a strong payoff for the simulation community. Errors arise at all levels, and their effect should be "appropriately" incorporated: we have the CAD error in the geometric model description e_{CAD}; the geometrical error e_{Geom} in representing the surface by the grid (e.g., rectilinear approximation of curves); the error associated with the cell shape e_s that may or may not be strongly tied to the analysis problem and data (e.g., surface fitting versus stress analysis near a crack tip); and the error e_p arising from conditioning and effects of finite-precision arithmetic.

3.5. Parallel Computation and Unstructured Grids

Unstructured and, especially, adaptive grids obviously pose some special problems for parallel computation. Simply put, an initial unstructured grid may be partitioned to balance the computational and communication load according to some metric [6, 7], but subsequent refinement and coarsening will undoubtedly lead to a strong imbalance. Codes such as PARMETIS and ZOLTAN [8] are being developed to address these issues, but there are significant problems yet to be resolved. The problems are exacerbated when one considers coupled multiphysics applications on different domains that share an interface, as is the case in fluid/solid interaction problems.

Some key issues. The question of appropriate metrics for determining a partition appears to be inextricably tied to the nature of the problem being analyzed, the solution algorithm, and the computer hardware. Current approaches are based exclusively on the grid geometry and are thereby limited. In view of the first issue, the dynamic repartitioning code should have appropriate "rules" or a simple expert system to guide the frequency of repartitioning and weights to guide the partitioning. Neural nets trained for a given class of problems may be of some limited use here. More generally, this will require an additional layer of complexity for the partitioner to use the metrics of the first issue for the machine in question. The problem of generating the grid in parallel and then adaptively refining and dynamically repartitioning in parallel (while maintaining consistency of the data structure across processor interfaces) is complex and must be addressed. For multiphysics applications with loosely decoupled algorithms, different unstructured/adaptive grids may often be advantageous. Strategies for projecting coupled field variables between grids (e.g., velocity from grid A to grid B and temperature concentrations from grid B back to grid A). The point here is that certain properties of the solution such as cell-based mass conservation may need to be preserved by the projection to satisfy solver sta-

bility requirements [9]. While this is not specifically a grid-generation issue, it does implicitly impact grid generation, especially for moving-mesh problems where remeshing and interpolation/projection are an integral part.

As a parting comment, as we promote more complex simulations on high-resolution grids (and conditioning deteriorates), what confidence do we have in the results computed for billions of cells on machines with very limited representations of the real number system? Surely the experience of the 1960s with 32-bit machines that led to the adoption of 64-bit arithmetic in the CDC systems must be revisited here and appropriate actions taken at the hardware and algorithm levels.

4. A STATUS REPORT ON HEXAHEDRAL MESHING: CONTRIBUTED BY D. WHITE

4.1. Introduction

While the increasing acceptance of tetrahedral meshes in a broad applications space has led to a decreased demand for hexahedral meshes, many finite-element applications still prefer hexes due to their inherent properties: increased accuracy and space-filling efficiency. Still other codes require hexahedrons due to special physics or solver requirements. The main impediment to using hexahedrons is the difficulty in computing them. Automatic discretization of a domain into good-quality tetrahedrons is provable mathematically. In contrast, for hexahedral elements there is only a proof that provides an assurance that all ball regions can be filled with topologically hexahedral elements, but with no minimal guarantee of element quality. Given the need to generate meshes containing only hexahedrons, and the general lack of an acceptable automatic method to generate them, research is underway at Sandia National Laboratories and elsewhere to complete a toolkit of meshing approaches or schemes to accomplish the task [10]. This section of the article discusses the schemes commonly used to generate unstructured hexahedral meshes.

4.2. Automatic Schemes

Over the years, several algorithms have been proposed for automatically meshing CAD shapes with hexahedral elements. The most common methods are automatic decomposition, inside-out methods, advancing-front algorithms, and tetrahedral splitting. Automatic decomposition techniques attempt to automatically cut CAD parts into meshable regions similar to those in a typical manual meshing process. Several algorithms have been developed to guide decomposition, including the medial axis transformation (MAT) and feature-based decomposition. In MAT approaches, branch points in the medial surface are bisected to decompose CAD parts [11]. The resulting pieces are meshed with midpoint subdivision, sweeping, or other hexahedral primitives [12]. Figure 1*a* shows an example of a part that is meshed using automatic decomposition based on the medial axis transform and midpoint subdivision to mesh the decomposed parts [11].

The feature-based decomposition approach searches for features in the solid and attempts to dissect these from the model and use primitives or other meshing approaches to mesh the separate parts [13]. Both approaches to automatic decomposition run into problems when the resulting decompositions still contain parts that are

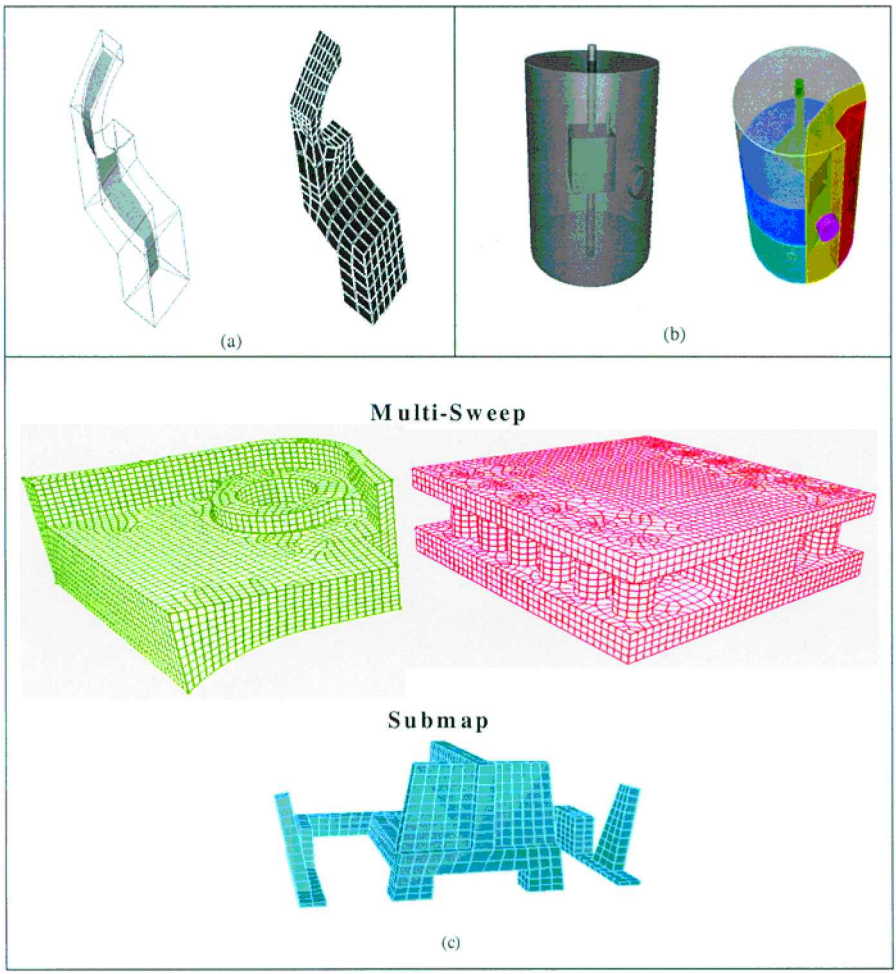


Figure 1. Hexahedral meshing: (*a*) medial surface and resulting hexahedral mesh; (*b*) manual decomposition of a 3-D object; (*c*) volumes meshed with the multisweep and submap methods.

not meshable. Both approaches lack the freedom a user would have in decomposing a part. An example of an assembly that requires creative decomposition is shown in Figure 1*b*. The cylindrical side appendage makes the part unsweepable along the major axis. As shown in the right of the figure, the smaller cylinder must have a sweep path cut through the larger material, while not interfering with the interior parts. Such decompositions rely on creative thinking, which is not captured by automatic decomposition techniques. Unfortunately, it is difficult to predict where automatic decomposition will work and where it will fail, making the algorithms less useful.

Inside-out methods start with inserting into a CAD model a mesh primitive such as a box or cylinder. The mesh is then grown or pruned until it conforms to the boundary of the solid. The nodes of the elements are relaxed to the surface of the geometry, and clean-up operations such as pillowing [14] are performed to improve

the quality of the mesh [14]. Given an unlimited element budget, this method is robust for most geometry types. Unfortunately, this method has two important drawbacks. First, the mesh on the boundaries of the surfaces can neither be predicted nor prescribed. This means that matching of meshes across part boundaries is either randomly achieved or is achieved by adding constraint equations. Second, the worst-quality elements generally appear on the boundary, which for many applications is where the highest-quality elements are required. There are many applications, however, where single parts are analyzed and the boundaries are of little concern, making this algorithm good for niche markets and as part of a toolkit of meshing algorithms.

Three-dimensional *advancing-front algorithms* such as whisker weaving [16, 17] and plastering [18] were once thought to be the solution to the hex-meshing problem. Plastering, a geometric advancing-front algorithm, was found to leave ribbon regions where no discernable hexahedral mesh could be inserted. Whisker weaving, a topological dual-space advancing-front method, was found to close reliably, but rarely resulted in meshes that were of a minimally acceptable quality. Despite their shortcomings, the contribution of both algorithms to hexahedral meshing is substantial. New automatic mixed-element meshing resulted from the plastering algorithm, where the void region is filled with pyramid elements on the interface and tetrahedrons on the interior voids. Whisker weaving inspired additional work such as the all-hex interface template, the geode [19]; and in other, more restricted uses of the algorithm [20].

Tetrahedral splitting is perhaps the most reliable and general form of automatic hexahedral meshing and is also the simplest. It takes the tetrahedral mesh of an object, which can be obtained automatically, and splits each tetrahedron into four hexahedrons by placing a node at the center of each triangle, edge, and tetrahedron and reconnecting the new edges formed by these nodes. The result is a fully conforming hexahedral mesh. The drawback to this meshing scheme is that its quality is usually not what is desired; even the best elements, those inside an equilateral tetrahedron, are of marginal quality. At worst, where slivers in the tetrahedral mesh may exist, the elements can be of extremely poor quality. Similar to the inside-out algorithm, for some applications this may be acceptable and provides an automatic approach to hexahedral meshing, and it is valuable in a niche application or as part of a toolkit.

4.3. Automated Methods

In the absence of a satisfactory high-quality automatic meshing scheme, many researchers have turned to improving the manual methods of generating hexahedral elements. Some of these improvements have come through extending sweeping and primitive algorithms, and development of automation control algorithms. In a manual approach a user will decompose a part into pieces that can be meshed with primitives or by sweeping. Mesh primitives are a set of predesigned meshes for typical or common shapes such as squares, triangles, and circles in two dimensions and cubes, tetrahedrons, and spheres in three dimensions. Sweeping essentially is an extension of a cylinder primitive in which the top circular surface mesh is extruded through the volume as hexahedrons. Sweeping requires that the "linking" surfaces or

sidewalls of the sweep axis be meshed with a structured or regular meshing scheme such as mapping [21].

Decomposing all the parts into primitives or sweeps is tedious and unnecessary. Preprocessing steps to perform a minimal amount of "pseudo"-decomposition automatically were added to extend sweeping and mapping. The mapping algorithm was extended to submapping, which uses virtual subdivision based on the boundary mesh to decompose a part into mappable subregions [22]. Sweeping was first extended to "pick up" additional source faces in the sweep as it progressed through the axis. The algorithm was then further extended to not only pick up faces but also terminate them as the sweep traveled along the axis [23]. This technique is referred to as "multisweep" or "Coopering." Examples showing the meshes produced by multisweep and submapping are shown in Figure 1*c*.

Primitives and sweeping often rely on user intervention to prescribe exact boundary intervals, surface meshes, and sweep directions. When meshing large assemblies of parts, managing and entering this data can become overwhelming, even for experienced users. Automation for controlling and relaxing the amount of user-supplied data has been another area of research in hexahedral meshing. Two algorithms that have substantially reduced this problem are automatic scheme selection and automatic interval assignment. Automatic scheme selection uses a sweepability proof to detect shapes that can be meshed with sweeping and other primitives [24]. The algorithm automatically assigns the proper surface schemes and assigns proper sweep directions. The automatic interval assignment algorithm solves a system of linear, integer constraint equations to provide proper edge intervals for meshing [25]. The constraints are based specifically on the requirements of the meshing algorithms that are to be used. Both automatic scheme selection and interval assignment greatly reduce the amount of data input required by sweeping and mesh primitives.

5. GENERAL GRID GENERATION: OCTREE METHODS AND MESH OPTIMIZATION: CONTRIBUTED BY G. A. HANSEN WITH W. R. OAKES, R. P. WEAVER, AND M. L. GITTINGS

5.1. Introduction

Mesh generation and mesh dynamics are important topics for the solution of large multidimensional problems in computational heat transfer (CHT). Advances in CHT methodology, along with the recent availability of powerful computational platforms, have made it feasible to consider detailed calculations on geometrically complex domains in two and three dimensions. Furthermore, the mesh must capture both the desired geometric and transient solution features within the computational domain. For problems with complex transient physics interacting with intricate geometric objects such as combustion modeling near a burner, traditional structured and unstructured mesh-generation approaches are often intractable. Generation of an appropriate mesh to support calculations of this complexity is a challenging, multidisciplinary problem; it is currently often more time and labor consuming to create the supporting mesh than to define and perform the desired simulation.

This discussion explores two topics in mesh generation: octree mesh generation and mesh optimization. The adaptive octree mesh-generation approach seeks a compromise between traditional mesh quality criteria and the ability to generate

automatically a mesh that captures relevant geometric and solution detail. The octree approach is statically and dynamically effective for arbitrarily complex problems, but often does not provide the best mesh for a typical simulation application near geometric boundaries. The second topic presented in this discussion examines a less flexible but more accurate approach (for a given cell size) of optimizing a traditional boundary-conformal mesh by optimizing the location of the mesh node points based on solution and geometric criteria. This adaptive method yields a high-quality mesh within certain constraints. This approach is less effective for large-displacement problems, where either topological refinement or mesh reconnection is better suited for the transient dynamics.

5.2. Adaptive Octree Mesh Generation

The adaptive octree mesh generation method seeks a compromise between mesh quality and the ability to discretize domains containing complex geometry in an automatic manner [26]. In practice, it is often the case that the generation of a typical structured or unstructured hexahedral mesh tends to be a very detailed, manual process for complex problems. For many problems, the geometry contained within the domain must be significantly simplified prior to mesh generation, both to facilitate the completion of the mesh in a reasonable period of time and to result in a mesh of acceptable quality for the desired simulation. If the target is a tetrahedral mesh, the mesh-generation process is more tractable. However, many applications cannot use tetrahedral elements effectively, and a significant amount of manual interaction is still necessary to develop such a mesh for a complex model.

When the simulation application has the flexibility to consider a nonconformal subdivision refinement strategy, is tolerant to nonaligned flow fields, and has the ability to resolve geometric boundaries explicitly, the octree mesh-generation method is a very effective solution to the trade-off between mesh-generation time and domain complexity.

The input data to the octree algorithm are straightforward, consisting of the physical extent of the computational domain and the objects contained within (e.g., Figure 2*a*). The minimum and maximum spatial resolutions of the mesh are specified by the user. Optionally, the user may supply initial conditions, guiding the algorithm to provide additional refinement in areas where it is needed to resolve solution length scales. This solution refinement condition may also be calculated within the generation application if it contains the governing equations; a pseudo-time step of the discrete system can be performed to calculate a metric that indicates where additional refinement is needed to resolve initial length scales present within the domain. Given the above input, the mesh-generation process generally proceeds to completion without any additional user input or guidance.

Given the input data, the octree mesh-generation method operates using a divide-and-conquer strategy. The basic approach begins with a discretization of the domain at the minimum level of refinement specified by the user, ignoring any geometry within the domain. Typically, this initial discretization is very coarse, and it is called the level-0 mesh (Figure 2*a*). Alternatively, better results may be obtained by boundary-fitting the level-0 mesh to bulk geometric features contained within the computational domain (ignoring those features whose size or impact metric is less

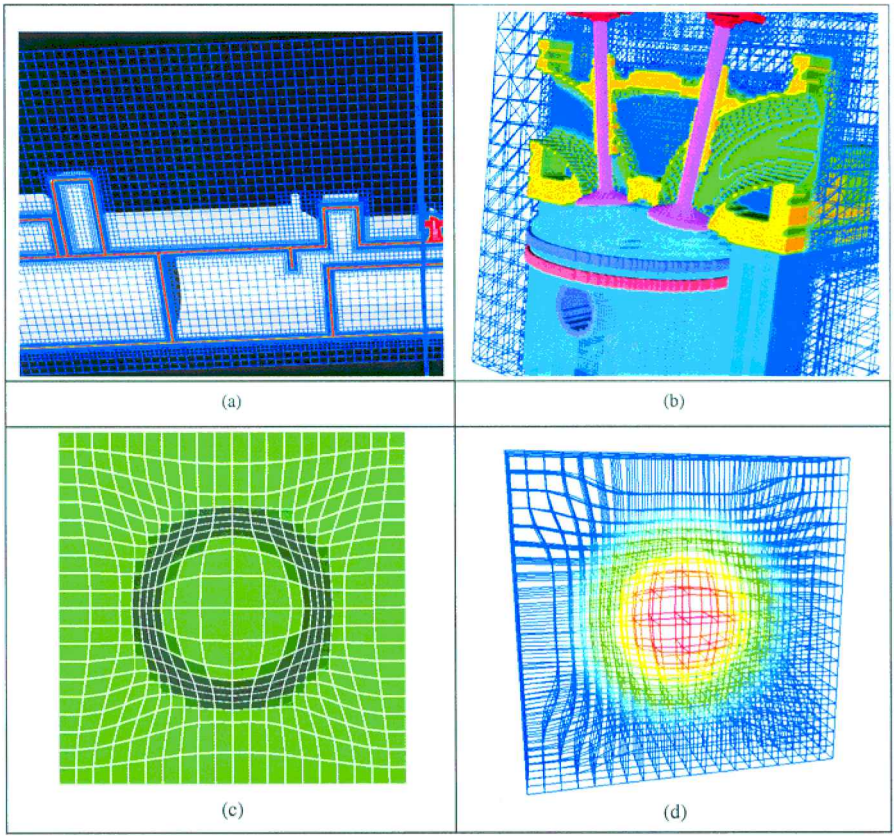


Figure 2. Examples of octree/quadtree mesh refinement methods and mesh optimization results. (*a*) A three-level geometric quadtree algorithm applied to the domain geometry using a simple Cartesian level-0 mesh. (*b*) An octree mesh for a reciprocating engine assembly. (*c*) A Cartesian initial mesh adapted to the solution of an arc-tangent hill problem. Darker colors indicate a larger magnitude of solution error. (*d*) Three-dimensional adaptation: a constant mesh metric composed with an error metric derived from the temperature of a point combustion problem.

than the spatial resolution of the level-0 mesh). The boundary-fitted approach will provide a more optimal global mesh, but it requires significant manual effort to obtain. The remainder of this discussion will assume that the first approach is used to create the level-0 mesh. With the base level-0 mesh, the adaptive octree approach adds spatial refinement to areas that require more discretization to capture either intricate geometric detail or small solution length scales. The refinement strategy is based on a spatial subdivision of those mesh cells that can benefit from additional discretization. The algorithm loops over the level-0 mesh, bisecting each candidate cell in each of the coordinate directions. Geometric octree refinement may be reduced to a spatial query operation as each cell is inspected, in turn, to determine if any geometric surfaces are contained within it. If this is the case, the cell is bisected. This algorithm may be applied recursively to achieve any measure of refinement necessary for the problem. Figure 2*a* shows an example of a three-level geometric

quadtree algorithm applied to the domain geometry using a simple Cartesian level-0 mesh.

To reiterate, the strength of the octree method lies in its ability to capture a disparity in spatial length scales in an automatic fashion. If the spatial query algorithm is developed in a general manner, the dimensionality and geometric complexity of the desired problem may be captured automatically, given simple input parameters. Figure 2*b* illustrates a cutaway diagram of an octree mesh generated on a reciprocating engine assembly.

These examples illustrate the flexibility of the octree approach in addressing geometric complexity in an automatic fashion. Adding solution adaptivity to the octree algorithm is also straightforward [27, 28]. The input data requirements, generating algorithm, and level of automation need not change as geometric and solution complexity increases.

5.3. Adaptive Mesh Optimization

A second method of dynamically optimizing a mesh to accommodate evolving solution details is to move mesh node points physically to more optimal locations. Traditionally, this form of mesh optimization involves a trade-off between the level of local adaptivity that may be tolerated without sacrificing the geometric integrity and smoothness of the mesh. Many approaches to providing solution adaptivity are derived by formulating an optimization problem targeted at minimizing an error metric over the mesh. These approaches may result in an overemphasis of solution criteria with respect to mesh geometry, resulting in a localized reduction in cell geometric quality. Ideally, one desires to couple the minimization of solution error with a system that optimizes the solution qualities of the mesh to provide adaptivity while maintaining a measure of mesh geometric quality. Consider the elliptic grid generation system, where $\mathbf{x}$ are the coordinates of the grid points in Cartesian 2-space [29, 30]:

$$g_{22}(\mathbf{x}_{\xi\xi} + P\mathbf{x}_{\xi}) + g_{11}(\mathbf{x}_{\eta\eta} + Q\mathbf{x}_{\eta}) = 0$$

where

$$P = -\frac{\mathbf{x}_{\xi} \cdot \mathbf{x}_{\xi\xi}}{g_{11}} - \frac{\mathbf{x}_{\xi} \cdot \mathbf{x}_{\eta\eta}}{g_{22}} = -\frac{1}{2}\frac{(g_{11})_{\xi}}{g_{11}} - \frac{(g_{12})_{\eta}}{g_{22}} + \frac{1}{2}\frac{(g_{22})_{\xi}}{g_{22}}$$

$$Q = -\frac{\mathbf{x}_{\eta} \cdot \mathbf{x}_{\eta\eta}}{g_{22}} - \frac{\mathbf{x}_{\eta} \cdot \mathbf{x}_{\xi\xi}}{g_{11}} = -\frac{1}{2}\frac{(g_{22})_{\eta}}{g_{22}} - \frac{(g_{12})_{\xi}}{g_{11}} + \frac{1}{2}\frac{(g_{11})_{\eta}}{g_{11}}$$

If one considers the spatial metric terms g_{ij} as prescriptive terms, it is possible to adjust the geometric characteristics of the mesh. Furthermore, g_{11} prescribes the mesh spacing in the first coordinate direction, g_{22} the second, and g_{12} is a measure of local grid orthogonality. If the metric surface g_{ij} is expressed as a composition of both mesh spatial information and a solution error metric, the composite metric will prescribe a mesh spacing that reflects the geometric characteristics of the previous mesh adjusted by the solution data. For example, Figure 2*c* shows the results of

combining a spatially constant initial mesh metric with an error metric derived from a hyperbolic tangent function.

This figure illustrates the actual physical mesh produced for this 2-D example problem. Visually, the darker cells in the diagram correspond to the physical region where the simulation error is higher. Detailed study of this result reveals an interesting interaction between the optimization of the mesh geometry and the adaptation of the mesh to the solution feature, particularly at points 45° from the horizontal and vertical axes. In this region, the requirements of the geometric problem and error problem are orthogonal; the g_{12} term of the geometric solution is driving the cell-included angles to 90°, and the solution g_{12} term is seeking included angles of 0° and 180°. Clearly, the incorporation of solution adaptation entails a compromise in traditional geometric quality criteria if the mesh topology cannot be modified appropriately. Figure 2*d* reveals similar behavior on a hypothetical 3-D point combustion problem; the coloration indicates the temperature of regions of the sphere as the flame propagates.

This dynamic adaptation approach appears very promising in application. However, there are two main areas of development needed before the approach is fully effective. The relative weight of the solution and geometric metrics on the composite problem are controlled by a user-specified parameter; the adaptation weight is problem specific and is a transient function. It may be possible to calculate this weighting function implicitly by seeking a functional convergence that distributes the solution error equally across the mesh. The approach and mechanics for achieving this result is an open topic. Second, the solution error is likely sensitive to the local topology of the mesh (the relative alignment of the mesh with the primary solution propagation direction influences the local error). Work remains both to quantify this error and to develop the mechanics to locally disconnect and reconnect mesh edges (and faces) to adapt the mesh graph dynamically to the transient solution.

6. ISSUES RELATED TO VISCOUS GRIDS: CONTRIBUTED BY Y. KALLINDERIS

6.1. Introduction

There is an ever-increasing demand to perform flow simulations that incorporate the complete details of geometry as well as sophisticated field physics. The success of numerical flow simulators depends to a great extent on the computational grid that is employed. As a consequence, grid generation has become a task of primary importance. Structured meshes consisting of blocks of hexahedra and unstructured grids consisting of tetrahedra have been the traditional means of discretizing 3-D flow domains. Hybrid grids usually consist of prisms and tetrahedra in three dimensions, and correspondingly quadrilaterals and triangles in two-dimensions. Layers of prisms are employed to resolve boundary layers and wakes, while tetrahedra cover the rest of the domain.

There are a number of issues to be addressed when dealing with turbulent flow simulations involving complex geometries. These considerations include: (1) the different orientation of the viscous flow features, (2) the disparate length scales that need to be resolved within the same domain, (3) the requirements of the Navier–Stokes solvers.

Feature orientation. The main features that are encountered in flow fields include boundary layers, wakes, shock waves, and vortices. These features have different orientations, making generation of a single grid that conforms to them very difficult. In addition, the mesh has to follow the boundaries of the computational domain. A hybrid grid which combines elements of differing orientation appears to be much more flexible in conforming to the flow features. The prisms are assigned the task of capturing the features that are following the body surface, while the tetrahedra are used for the features that are away from the boundaries (e.g., shocks and vortices).

Disparate length scales. The different spatial scales encountered in viscous flows vary by orders of magnitude from each other, scales imposed by the flow features and geometry. The laminar sublayer requires placement of grid points at distances away from the wall of the order of one-millionth the scale of the geometry, while the points at the far field may be at a distance of order 1 from one another. Shock waves and vortices have very different scales as well. Furthermore, the details of the geometry frequently impose scales on the grid generator. The gaps between the main wing and the flap and the tip clearances in turbomachinery geometries are typical examples of small scales.

The issue becomes even more complex when taking into account the directionality of the different scales. The small scale required in the boundary layers is in the direction normal to the surface, while much larger sizes of the mesh are sufficient in the lateral directions. Similar directionality also exists in wakes and shock waves. This directionality leads to the issue of generating high-aspect-ratio grid cells. Generation of thin prismatic grids for the boundary layers and wakes has the advantage of being feasible, fast, and resulting in a smaller number of elements compared to tetrahedra. On the other hand, the isotropic nature of tetrahedra appears to be appropriate for the vortices and other regions of the domain where the flow is changing equally in all directions.

Navier–Stokes solver requirements. Navier–Stokes solvers place strict requirements on the mesh. Accuracy and stability of the numerical methods depend crucially on the local resolution and the uniformity of the grid. Smooth transition of element sizes at the prism/tetrahedra interface is important for accuracy and robustness of Navier–Stokes numerical methods. Furthermore, computing resources, in terms of CPU time and memory storage, are dictated by the number of grid elements.

Employment of the thin semistructured prismatic elements in the regions of shear layers results in sufficient accuracy with significantly reduced computing resources compared to all-tetrahedral meshes. The flow field on the body surface usually contains regions of strong flow directionality, such as the leading and trailing edges of a wing. Generation of anisotropic surface grid elements results in significant savings in the number of elements without sacrificing accuracy.

6.2. Traditional Types of Grids in Three Dimensions

Structured meshes consisting of hexahedra and unstructured meshes consisting of tetrahedra have been the traditional means of discretizing 3-D Eulerian

flow domains. Both approaches have been challenged in recent years, as applications move to large-scale turbulent flows with very complex geometries. The two main kinds of structured meshes are multiblock and overset grids, which are generated with elliptic, hyperbolic, or interpolation types of methods. Unstructured tetrahedral meshes are generated via the advancing front, Delaunay, and octree types of approaches.

Structured multiblock meshes. In the multiblock approach, the computational domain is divided into several subdomains (blocks). Separate hexahedral grids are then generated within each block. The grid lines at the block interfaces may be continuous (composite grids) or discontinuous (patched). These grids have the simplicity of their inherent structure, resulting in simple data structures that do not require a lot of computer memory. Furthermore, implementation of the corresponding numerical solvers and flow visualization tools are relatively simple.

In the case of complex geometries, a large number of blocks needs to be defined by the user. Definition of the blocks and their interfaces becomes increasingly difficult and time consuming as the number of blocks increases. Furthermore, an experienced user is needed to perform this task. Steps to remove the user from the process and to increase automation have been taken in the past few years via appropriate graphical user interfaces and the development of automatic blocking procedures.

Overset grids. The difficulties associated with interfacing a large number of blocks of different orientation and sizes led to the development of overlapping structured grids. This method is also known as *chimera* mesh generation. Structured grids are generated independently for each component of a complex geometry, and the grids are overset on each other. An example of the flexibility of the method is its employment for simulation of the flow around the Space Shuttle. A rather complicated data structure is required to facilitate the transfer of information between the different overlapping meshes. The generation of the grids is not entirely independent of each other, since overlapping of meshes of very different resolution should be avoided due to the large interpolation errors. Another complication with this approach is maintenance of conservation of the flow quantities in the regions of overlapping, which is crucial to computation of flows with shock waves.

Unstructured grids. A radical alternative to a structured mesh is the use of tetrahedra. Tetrahedral grids provide flexibility in 3-D grid generation, since they can cover complicated topologies more easily than the hexahedral meshes. The lack of structure allows complex geometries to be discretized in a single block. The three main approaches of unstructured grid generation are *Delaunay methods*, *advancing-front methods*, and *octree-based* techniques.

Tetrahedral grid generators have focused on producing valid grids for complex domains. This approach has been quite successful for the case of inviscid flow simulations described by the Euler equations. However, the demands on level of accuracy and computing resources of Navier–Stokes computations have been enormous for tetrahedral element-based solvers. Turbulent flow simulations around a single wing can easily employ 10 million tetrahedra and on the order of a gigaword of memory. Also, generation of tetrahedral cells for boundary layers is difficult. In these regions the main solution gradients occur in the direction normal to the

surface, which requires high-aspect-ratio cells. It appears that structured grids are superior in capturing the directionality of the flow field over such viscous regions.

Cartesian grids. The meshes we have seen so far follow the geometry of the surfaces involved. A radical alternative to body-conforming meshes are the so-called *Cartesian grids*. These are generated ignoring the presence of the bodies and are aligned with the Cartesian coordinates. A master hexahedron encompassing the body is recursively subdivided to create hexahedral elements that become progressively smaller as the surface is approached. Generation is simple and automatic. The hexahedra can be further divided adaptively based on the curvature of the surface and/or the solution gradients. Basically, grid generation is not an issue for very complex geometries.

The most serious problem of the Cartesian approach is the poor quality of the mesh close to the surface. The hexahedra intersect the boundaries in a random fashion, and control of the shapes and sizes of these elements is difficult. The elements are cut by the boundaries in several different ways, requiring special implementation of the boundary conditions. Special care also is required in order to avoid the small time steps that are normally required to integrate extremely small elements that are cut by the boundaries. The method has been applied to simulate primarily potential and inviscid flows. Viscous flow simulations are difficult to perform with Cartesian grids.

6.3. Mixed-Element Grids

Employment of a single type of grid appears to be insufficient in resolving 3-D viscous flow domains accurately and efficiently. A relatively new approach has been the use of hybrid meshes that provide considerable flexibility. Hybrid grids consisting of prisms and/or hexahedra as well as tetrahedra combine the advantages of both structured and unstructured approaches. A typical hybrid grid generator consists of two major parts: (1) the prisms or hexahedra generator, an algebraic, marching-type technique, and (2) the tetrahedra generator.

The structured marching method for prisms and hexahedra. An unstructured triangular grid is employed as the starting surface to generate a prismatic mesh. This grid, covering the body surface, is marched away from the body in distinct steps, resulting in generation of semistructured prismatic layers in the marching direction. If the surface is discretized with quadrilaterals, then a hexahedral mesh will be created with the same marching method.

The process can be visualized as a gradual inflation of the body's volume. A major issue with marching methods is to avoid crossing of the grid lines. There are three main aspects to the algebraic grid generation process: (1) determination of the directions along which the nodes will march (marching vectors), (2) determination of the distance by which the nodes will march along the marching vectors, and (3) smoothing operations on positioning of the nodes on the new layer.

Each node on the marching surface is advanced along a marching vector. The marching direction is based on the *node manifold*, which consists of the group of faces sharing the node to be marched. The primary criterion to be satisfied when marching is that the new node should be visible from all the faces on the manifold

(the *visibility condition*). Determination of marching distances is based on the characteristic angle of the manifold of each node to be marched. The average marching step for each layer is computed based on a user-specified initial marching step on the body surface and a stretching factor.

The initial marching vectors are the normal vectors. However, this may not provide a valid grid since overlapping may occur, especially in concave regions of the grid surface with closely spaced nodes. To prevent overlapping, the directions of the marching vectors must be altered. Altering of the directions should not end abruptly in the local neighborhood of the nodes involved, since this may cause overlapping in nearby regions. A gradual reduction of the magnitude of the change in the vector direction is accomplished via a number of weighted Laplacian-type smoothing operations over the marching vectors of all nodes. A similar procedure is employed for the smoothing of the marching steps to eliminate abrupt changes in cell sizes.

Typical Navier–Stokes integration methods impose restrictions on the spacing of the points along the marching lines and on the smoothness of these lines. In other words, the prismatic grid should not be excessively stretched or skewed. Constraints are imposed on the lateral and normal distribution of marching step sizes and the deviation of the direction of the marching vectors from one layer to the next. The above constraints reduce "kinks" in the marching vector directions as well as abrupt changes in step sizes, thus providing a smooth mesh suitable for viscous flow computations. Since the visibility criterion is the ultimate test for the validity of the mesh, this criterion is the final constraint that is imposed on the grid.

A mixed octree/advancing-front method for tetrahedra. A combined octree/advancing-front method is used to generate the unstructured grid. The advancing-front type of methods require specification by the user of the distribution of three parameters over the entire domain to be discretized. These field functions are (1) node spacing, (2) grid stretching, and (3) direction of the stretching. Using the octree/advancing-front method, these parameters do not need to be specified. Instead, they are determined via an automatically generated octree.

The octree is constructed via a *divide-and-conquer* process starting with a *master* hexahedron that contains the body. This hexahedron is recursively subdivided into eight smaller hexahedra called *octants*. Any octant that intersects the body is a *boundary octant* and is subdivided further (inward refinement). The subdivision of a boundary octant ceases when its size matches a local length scale. The choice of the local length scale depends on the particular application of the octree. The length scale can be chosen to be local prism thickness, surface edge length, or surface curvature. For hybrid prismatic/tetrahedral mesh generation, the local length scale is simply the local thickness of the last prismatic layer. This will ensure that the size of the tetrahedra in the direction normal to the outer prismatic surface is the same as the height of the neighboring prisms. This smooth transition in size from the prisms to the tetrahedra is important for accuracy of the numerical method.

Two important features of the octree/advancing-front method are its ability to match disparate length scales and its geometry independence. The octree is able to ensure a smooth size transition over the large range of length scales present in a

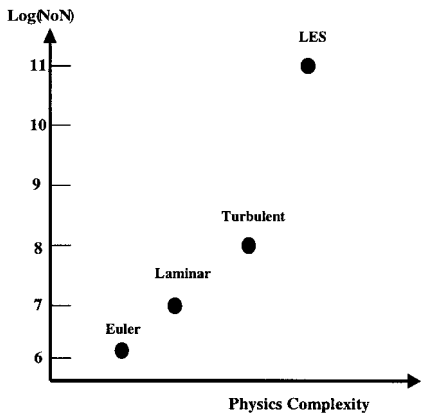
viscous mesh. The octree may be used for many different types of geometries with minimal user interaction.

7. NEXT-GENERATION LARGE-SCALE AEROSPACE SIMULATIONS ON UNSTRUCTURED GRIDS: CONTRIBUTED BY N. WEATHERILL WITH O. HASSAN, K. MORGAN, J. W. JONES, B. G. LARWOOD, AND K. SORENSON

7.1. Introduction

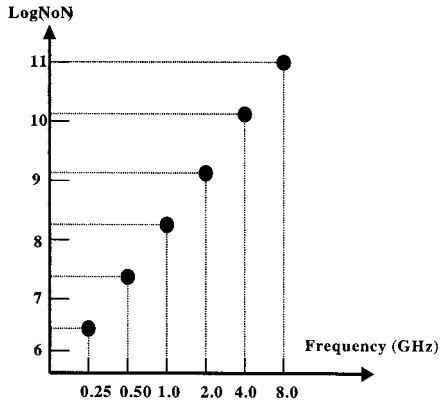
The advent of the vector supercomputers of the 1970s, of which the CRAY 1S is perhaps the most famous, had a major influence on scientific simulation. The technology acted as a catalyst for new algorithms that were able to address emerging and challenging applications. In the last few years, the next generation of computers has emerged in the form of massively parallel computer hardware. Again there is an opportunity for the simulation community to utilize this new computer power to open new avenues for research and to attempt real-world simulations that in the past were out of the range of all but a handful of extremely expensive computers. In the areas of computational aerodynamics and electromagnetics (focused primarily on radar cross section), it is clear from our estimates (see Figures 3*a* and 3*b*) that very large computational grids will be required for future simulations. However, with parallel computer platforms and suitable software, the next generation of simulations is feasible (see Figure 3*c*).

Some years ago, therefore, we embarked on a long-term research program to enhance our software capability in computational fluid dynamics (CFD) and computational electromagnetics (CEM) so as to provide the necessary basis for the next generation of simulations. It was deemed necessary to parallelize all the different steps in the computational cycle, from geometry input to unstructured mesh generation, to simulation, to visualization and data mining, and on to mesh adaptation. To aid in the usability of such software a parallel simulation user environment

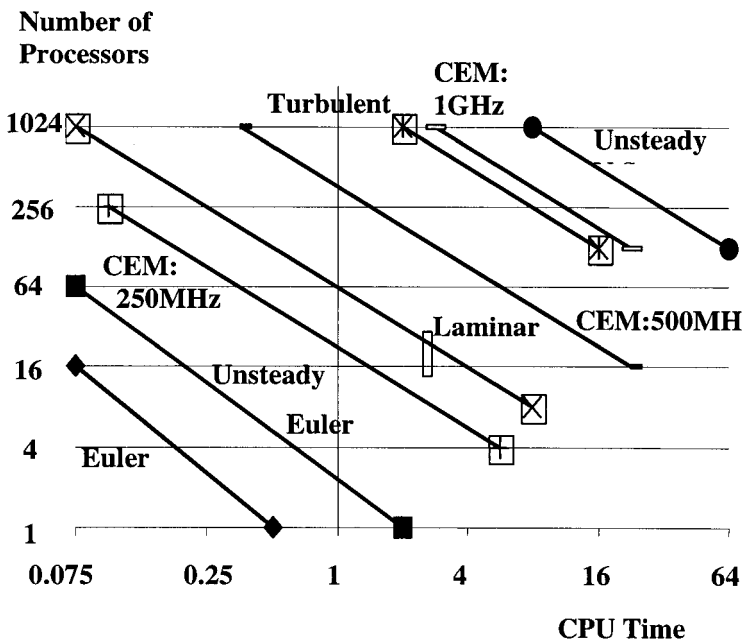


(a)

Figure 3. (*a*) Mesh requirements for computational fluid dynamics for a complete aircraft.



(b)



(c)

Figure 3. (*b*) Mesh requirements for the simulation of electromagnetic scatter from an aircraft 20 m in length (NoN = number of nodes). (*c*) Estimated computing time in hours required for simulations in the next five years.

(PSUEII), within which all the parallel modules were embedded, was also developed. Details of some of the developments and performance of these modules have been reported previously [31–35]. However, a very brief summary will be given.

Grid generation. Following a surface triangulation of the outer boundary of the domain, a geometric decomposition is constructed which subdivides the domain into an arbitrary number of partitions. The interdomain boundaries are meshed using a 2-D unstructured mesh generator and the closed subdomains are then farmed out to individual processors on which the volume mesh of tetrahedra is constructed. Communication between the manager and the worker processors is performed using MPI (the Message Passing Interface). To improve computational efficiency, dynamic load balancing is used.

Simulation. The solver technology used for both fluids (Euler or Navier–Stokes equation) and electromagnetics (Maxwell's equations) is finite element based with explicit time integration. The basic data structure used in the solvers is edge based, where the unknown variables are defined at nodes of the mesh. Communication between neighbor subdomains is defined so that all operations can be performed along edges. Edges are not duplicated, but common interface boundary nodes are stored in neighboring subdomains. MPI is used for communication between processors.

Adaptation. Given a solution on an initial mesh, *h*-refinement can be employed to provide additional resolution where indicated by an error estimator. Mesh refinement is applied in the different subdomains, with care taken to ensure consistency of the grid should refinement be required along interface boundaries. Nodes added on the configuration geometry are taken back to the original surface, requiring details of the geometry to be sent to the different processors.

Visualization. The basis of the parallel visualization toolkit is for all the searching and computationally intensive work to be performed on the processors and only the data required for rendering sent to the workstation for visual display. A set of library routines has been developed to handle the communication. Fast search routines operating between subdomains, and hence held on different processors, have been developed that utilize octree data structures. Given geometry, the visualization, meshing, simulation, adaptation, and postprocessing are all achieved in parallel without a requirement, at any stage in the cycle, to bring together, within one domain, the simulation data. As such, it is our premise that no computational bottlenecks are created. Given this development, the new software and hardware technology can be fully exercised.

7.2. Engineering Simulation

As an example of a large-scale simulation, the propagation of a single wave through an engine duct is considered (see Figure 4). Maxwell's equations, written in the time domain, are solved using an explicit time integration procedure. The dimensions of the duct are 6.2 m by 0.45 m by 0.45 m. The incident wave is 3 cm, which gives a frequency of 10 GHz. If it is assumed that 10 grid nodes are required per wavelength, then the number of nodes required in the mesh is $210 \times 15 \times 15 \times 10^3$, or approximately 47 million nodes. To generate a uniform mesh with the required size, background spacing was set at 3 units. The details of the mesh generated are given in Table 1. The time required to simulate one cycle of the wave, using 32 processors, was 8 h. To complete the simulation on a machine with 1024 processors would take approximately 4.5 days. The mesh and solution data were then passed back into the

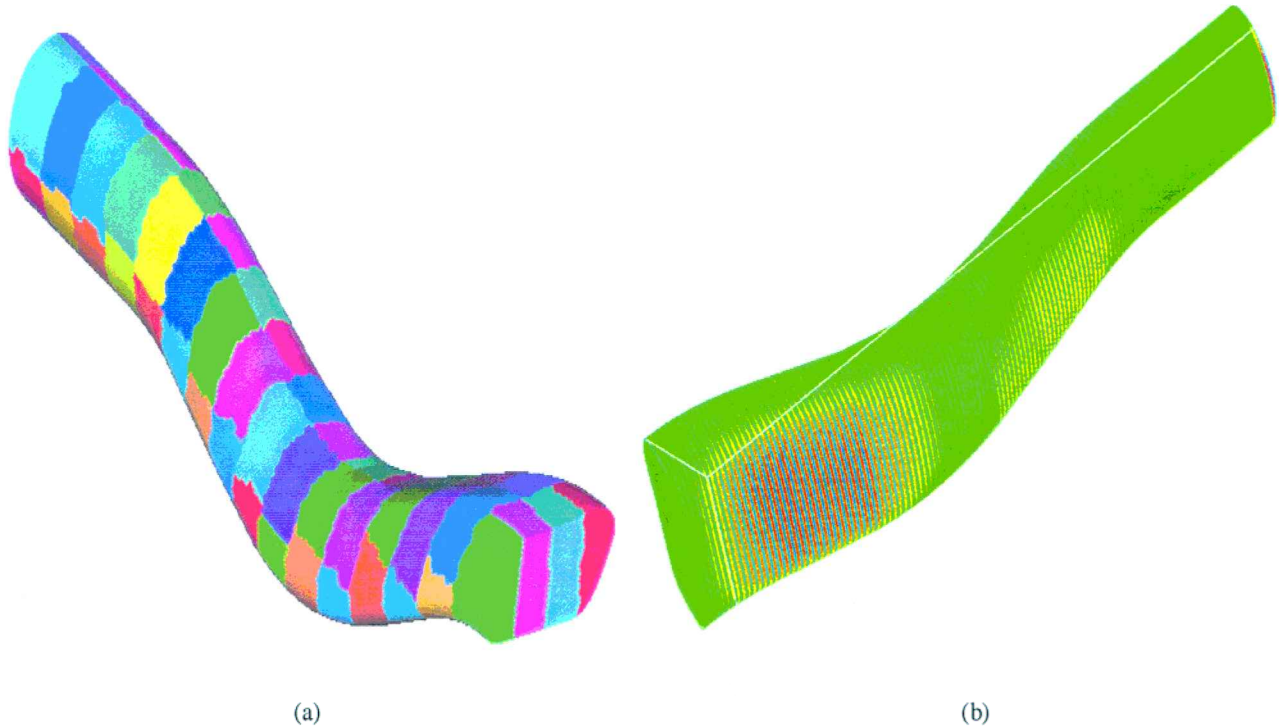


Figure 4. Details of the parallel decomposition and solution for a computational electromechanics simulation of an aerospace duct: (*a*) 128 partitions; (*b*) results after one cycle.

Table 1. Statistics of the unstructured mesh used in the computation shown in Figure 4

Geometry	
No. of surfaces	94
No. of curves	188
Mesh	
No. of partitions	128
No. of processors (R12,000 400 MHz)	24
Surface mesh	
No. of nodes	1,834,328
No. of triangles	3,668,652
Time for generation	1.6 h
Volume mesh	
No. of nodes	44,078,548
No. of tetrahedra	236,356,076
Size of volume mesh file	5.1 GB
Size of communication data file	0.23 GB
Time for generation	18 h

PSUE II and rendered using the parallel visualization technology to produce Figures 4*a* and 4*b*. Figure 4*a* shows the mesh elements colored by the partition in which they reside, and Figure 4*b* shows the wave propagation after one cycle.

8. SUMMARY

We have provided only a brief glimpse into the world of grid generation as it stands today. There are many sources of additional information concerning grid generation and associated issues. A very small sampling of these might include reference books such as the *Handbook of Grid Generation* [36], textbooks by Carey [37], Knupp [38], and Frey and George [39], and research articles or summaries as prepared by Teng and Wong [1], for example. A ready source of introductory material plus a host of additional World Wide Web links for further information can be found at Websites such as those of Steve Owen for the Meshing Research Corner at http://www.andrew.cmu.edu/user/sowen/mesh.html, of Robert Schneiders at http://www-users.informatik.rwthaachen.de/ ~roberts/meshgeneration.html, of David Epstein for Geometry in Action at http://www.ics.uci.edu/~epstein/geom.html, and at Nina Amenta's Directory of Computational Geometry Software at http://www.geom.umn.edu/software/cglist.

The contributors to this article wish to extend our thanks to Drs. Graham de Vahl Davis and Eddie Leonardi, organizers of CHT'01, for graciously providing the encouragement to proceed and the time in the full schedule for this panel discussion. It was our pleasure to participate in it.

REFERENCES

1. S.-H. Teng and C. W. Wong, Unstructured Mesh Generation: Theory, Practice, and Perspectives, *Int. J. Comput. Geom. Appl.*, vol. 10, pp. 227–266, 2000.

2. G. F. Carey, A Mesh Refinement Scheme for Finite Element Computations, *J. Comput. Meth. Appl. Mech. Eng.*, vol. 7, pp. 93–105, 1976.
3. G. F. Carey, Keynote Lecture, Adaptive Techniques and Related Issues in Finite Element Modeling of Heat and Fluid Flow, ICHMT Symposium (CHT'01), Advances in Computational Heat Transfer, Palm Cove, Cairns, Queensland, Australia, May 2001.
4. P. Knupp, Achieving Finite Element Mesh Quality via Optimization of the Jacobian Matrix Norm and Associated Quantities. Part II—A Framework for Volume Mesh Optimization and the Condition Number of the Jacobian Matrix, *Int. J. Numer. Meth. Eng.*, vol. 48, pp. 1165–1185, July 2000.
5. S. Iqbal and G. F. Carey, Neural Nets for Mesh Assessment, TICAM Report #02-02, University of Texas at Austin, Austin, TX, January 2002.
6. B. Hendricksen and R. Leland, Multidimensional Spectral Load Balancing, Tech. Rep., SAND 93-0074, Sandia Natl. Lab., January 1993.
7. G. Karypis and V. Kumar, A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs, *SIAM J. Sci. Comput.*, vol. 20, pp. 359–392, 1998.
8. B. Hendrickson and K. Devine, Dynamic Load Balancing in Computational Mechanics, *Comput. Meth. Appl. Mech. Eng.*, vol. 184, pp. 485–500, 2000.
9. G. F. Carey, G. Bicken, V. Carey, C. Berger, and J. Sanchez, Locally Constrained Projections, *Int. J. Numer. Meth. Eng.*, vol. 50, pp. 549–577, 2001.
10. S. A. Mitchell, Project Leader of the CUBIT Mesh Generation Toolkit, http://endo.sandia.gov/cubit.
11. C. G. Armstrong, D. J. Robinson, R. M. McKeag, T. S. Li, S. J. Bridgett, R. J. Donaghy, and C. A. McGleenan, Medials for Meshing and More, *Proc. 4th Int. Meshing Roundtable*, SAND95-2130, pp. 277–288, October 1995.
12. A. Sheffer, M. Etzion, A. Rappoport, and M. Bercovier, Hexahedral Mesh Generation Using the Embedded Voronoi Graph, *Proc. 7th Int. Meshing Roundtable*, SAND98-2250, pp. 347–364, October 1998.
13. Y. Lu, R. Gadh, and T. J. Tautges, Volume Decomposition and Feature Recognition for Hexahedral Mesh Generation, *Proc. 8th Int. Meshing Roundtable*, SAND99-2288, pp. 269–280, October 1999.
14. S. A. Mitchell and T. J. Tautges, Pillowing Doublets: Refining a Mesh to Ensure that Faces Share at Most One Edge, *Proc. 4th Int. Meshing Roundtable*, SAND95-2130, pp. 231–240, October 1995.
15. R. Schneiders, R. Schindler, and F. Weiler, Octree-Based Generation of Hexahedral Element Meshes, *Proc. 5th Int. Meshing Roundtable*, SAND96-2301, pp. 205–216, October 1996.
16. T. J. Tautges, T. Blacker, and S. A. Mitchell, The Whisker Weaving Algorithm: A Connectivity-Based Method for Constructing All-Hexahedral Finite Element Meshes, *Int. J. Numer. Meth. Eng.*, vol. 39, pp. 3327–3349, 1996.
17. N. T. Folwell and S. A. Mitchell, Reliable Whisker Weaving via Curve Contraction, *Proc. 7th Int. Meshing Roundtable*, SAND98-2250, pp. 365–378, October 1998.
18. T. D. Blacker and R. J. Meyers, Seams and Wedges in Plastering: A 3D Hexahedral Mesh Generation Algorithm, *Eng. Comput.*, vol. 2, pp. 83–93, 1993.
19. S. A. Mitchell, The All-Hex Geode-Template for Conforming a Diced Tetrahedral Mesh to Any Diced Hexahedral Mesh, *Proc. 7th Int. Meshing Roundtable*, SAND98-2250, pp. 295–305, October 1998.
20. M. Muller-Hannemann, Hexahedral Mesh Generation by Successive Dual Cycle Elimination, *Proc. 7th Int. Meshing Roundtable*, SAND98-2250, pp. 365–378, October 1998.
21. M. Whitely, D. R. White, S. E. Benzley, and T. Blacker, Two and Three-Quarter Dimensional Meshing Facilitators, *Eng. Comput.*, vol. 12, pp. 155–167, December 1996.

22. D. R. White, L. Mingwu, S. E. Benzley, and G. D. Sjaardema, Automated Hexahedral Mesh Generation by Virtual Decomposition, *Proc. 4th Int. Meshing Roundtable*, SAND95-2130, pp. 165–176, October 1995.
23. T. Blacker, The Cooper Tool, *Proc. 5th Int. Meshing Roundtable*, SAND96-2301, pp. 13–30, October 1996.
24. D. R. White and T. J. Tautges, Automatic Scheme Selection for Toolkit Hex Meshing, *Int. J. Meth. Eng.*, vol. 49, pp. 127–144, September 2000.
25. S. A. Mitchell, High Fidelity Interval Assignment, *Proc. 6th Int. Meshing Roundtable*, SAND97-2399, pp. 33–44, October 1997.
26. W. R. Oakes, P. J. Henning, M. L. Gittings, and R. P. Weaver, On 3D, Automated, Self-Contained Grid Generation within the RAGE CAMR Hydrocode, in B. K. Soni, J. Hauser, J. F. Thompson, and P. Eiseman (eds.), *7th Int. Conf. Numerical Grid Generation in Computational Field Simulation*, pp. 973–981, September 25–28, 2000.
27. R. M. Baltrusaitis, M. L. Gittings, R. P. Weaver, R. F. Benjamin, and J. M. Budzinski, Simulation of Shock-Generated Instabilities, *Phys. Fluids*, vol. 8, pp. 2471–2483, 1996.
28. R. P. Weaver, M. L. Gittings, M. R. Clover, and H. P. Pritchard, The Parallel Implementation of RAGE: A 3-D Continuous Adaptive Mesh Refinement Shock Code, ISSW22, July 18–23, 1999.
29. A. Khamayseh and G. Hansen, Quasi-Orthogonal Grids with Impedance Matching, *SIAM J. Sci. Comput.*, vol. 22, pp. 1220–1237, 2000.
30. J. F. Thompson, Z. U. A. Warsi, and C. W. Mastin, *Numerical Grid Generation, Foundations and Applications*, Elsevier, New York, 1985.
31. K. Morgan, P. J. Brookes, O. Hassan, and N. P. Weatherill, Parallel Processing for the Simulation of Problems Involving Scattering of Electromagnetic Waves, *Comput. Meth. Appl. Mech. Eng.*, vol. 152, pp. 157–174, 1998.
32. R. Said, N. P. Weatherill, K. Morgan, and N. A. Verhoeven, Distributed Parallel Delaunay Mesh Generation, *Comput. Meth. Appl. Mech. Eng.*, vol. 177, pp. 109–125, 1999.
33. K. Morgan, N. P. Weatherill, O. Hassan, P. Brookes, R. Said, and J. W. Jones, A Parallel Framework for Multi-Disciplinary Aerospace Engineering Simulation Using Unstructured Meshes, *Int. J. Numer. Meth. Fluids*, vol. 31, pp. 159–173, 1999.
34. M. T. Manzari, O. Hassan, K. Morgan, and N. P. Weatherill, Turbulent Flow Computations on 3D Unstructured Grids, *Finite Elements Anal. Des.*, vol. 30, pp. 353–363, 1998.
35. N. P. Weatherill, E. A. Turner-Smith, M. J. Marchant, O. Hassan, and K. Morgan, An Integrated Software Environment for Multi-Disciplinary Computational Engineering, *Eng. Comput.*, vol. 16, pp. 913–933, 1999.
36. J. F. Thompson, B. K. Soni, and N. P. Weatherill (eds.), *Handbook of Grid Generation*, CRC Press, New York, 1999.
37. G. F. Carey, *Computational Grids: Generation, Adaptation and Solution Strategies*, Taylor & Francis, Washington, DC, 1997.
38. P. Knupp and S. Steinberg, *The Fundamentals of Grid Generation*, CRC Press, New York, 1993.
39. P. J. Frey and P.-L. George, *Mesh Generation: Application to Finite Elements*, Hermes, Oxford, U.K., 2000.